Proceedings of the 20th International Semantic Web Conference (ISWC) 2021:(submitted)
draft material -- please do not cite

Context-Enriched Learning Models for Aligning Biomedical Vocabularies in the UMLS Metathesaurus

No Institute Given

Abstract. The Unified Medical Language System (UMLS) Metathesaurus is a biomedical terminology integration system that integrates over 200 biomedical vocabularies. The Metathesaurus construction process mainly relies on lexical algorithms and manual curation. A lexical-based learning model (LexLM) was developed to predict synonymy among Metathesaurus terms. The LexLM largely outperforms a rule-based approach (RBA) that approximates the current construction process. However, the LexLM has the potential to be improved further because it only uses lexical information from the source vocabularies, while the RBA also takes advantage of contextual information.
In this paper, we develop context-enriched learning models (ConLMs) by adding to the LexLM multiple types of contextual information available to the UMLS editors, including source synonymy (SS), source semantic group (SG), and source hierarchical relations (HR). We represent contextual information in context-enriched knowledge graphs (ConKGs) with four variants ConSS, ConSG, ConHR, and ConAll. We train the embeddings for the ConKGs using different KG embedding techniques, including TransE, DistMult, HolE, and ComplEx. Finally, we create the ConLMs by concatenating the ConKG embedding vectors with the word embedding vectors from the LexLM. Our extensive experiments show a substantial performance improvement from the ConLMs over the LexLM, namely +4.41% in precision (93.16%), +2.27% in F1 (92.88%), while retaining similar performance in recall. Our experiments also show that the combination of TransE and ConHR performs best among the pairwise combinations of KG embedding techniques and ConKG variants. Finally, we show that adding contextual information not only benefits pairs of terms with high lexical similarity as we expected, but also improves performance on all generalization test datasets. These results demonstrate the importance of using contextual information in the UMLS vocabulary alignment.

1 Introduction

The Unified Medical Language System (UMLS) Metathesaurus is a biomedical terminology integration system developed by the US National Library of Medicine to integrate biomedical terms by organizing clusters of synonymous terms into concepts. The current construction process of the UMLS Metathesaurus heavily relies on lexical similarity algorithms to identify candidates for

synonymy. Additionally, synonymy asserted between terms in a source vocabulary is generally preserved in the Metathesaurus and terms that do not share a common semantics are prevented from being recognized as synonymous. Final synonymy determination comes from human curation by the Metathesaurus editors. Given the current size of the Metathesaurus with over 15 million terms from 214 source vocabularies, this process is inevitably costly and error-prone (as pointed out by [4, 5, 15, 21, 22].) Nguyen et al. [25] formalized synonymy prediction in the Metathesaurus as a vocabulary alignment problem (UVA). Additionally, to support performance evaluation of vocabulary alignment algorithms, they developed a rule-based baseline ("RBA baseline") that approximates the Metathesaurus construction process described above.

Motivation. To improve the current Metathesaurus construction process, Nguyen et al. [25] developed a supervised learning approach with lexical learning models (LexLMs) that largely outperformed the RBA baseline. However, they note as a limitation of their work that they only leveraged lexical information and did not include any contextual information. The LexLMs used a Siamese architecture [23] with BioWordVec embeddings [35] and LSTM layers [9] for computing semantic similarity scores between biomedical terms, and subsequently using these scores to predict synonymy. Moreover, they created different dataset variants with different degrees of lexical similarity among the negative examples for training the LexLMs and testing their generalization. Their experiments confirmed the hypothesis that the degree of lexical similarity among negative examples strongly influences the performance of the LexLMs. The LexLM variant trained on the most diverse dataset performs best across the four generalization datasets with different degrees of lexical similarity. Therefore, we will reuse this variant as the reference LexLM ("LexLM baseline") for our evaluation. Section 5.2 will describe the datasets used in this paper.

Although the LexLM largely outperformed the RBA, we believe that learning can be further improved, because the LexLM only uses lexical information from biomedical vocabularies, while the RBA also takes advantage of contextual information. More specifically, we hypothesize that adding contextual information to the LexLM will support the disambiguation of homonymous terms. In the example from Figure 1, the terms "COLD" from NCI and "Cold" from SNOMEDCT_US have the same word embeddings and will therefore be predicted as synonymous by the LexLM. However, these two terms can be disambiguated by taking into account their source synonyms, such as "Chronic Obstructive Lung Disease" and "Common cold", respectively. Similarly, disambiguation can be provided by hierarchically-related terms in the source vocabularies, such as the parent terms "Chronic Lung Disorder" and "Viral upper respiratory tract infection", respectively. Of note, source semantics (i.e., the semantic group assigned to the top-level terms in a source vocabulary and inherited by all descendant terms) cannot be used to distinguish between these two disease terms. We will use these two homonymous terms as our running example.

In this work, we develop context-enriched learning models (ConLMs) by adding contextual information to the lexical information in the LexLM to im-

prove synonymy prediction at scale in the UMLS Metathesaurus (UVA task). However, this is neither a systematic evaluation, nor a performance benchmark among all KG embedding techniques. Instead, our goal is to explore the use of KG embedding approaches to represent contextual information in the UVA task. More specifically, we explore candidate algorithms from the main families of KG embedding described in [28, 31], namely algorithms based on (a) transitional distance and (b) semantic matching.

Objectives. Our first objective is to improve the performance of the LexLM by adding contextual information. In practice, we develop context-enriched learning models (ConLMs) by adding to the LexLM multiple types of contextual information using four different KG embedding techniques. We hypothesize that the ConLMs will outperform the LexLM, particularly because the addition of contextual information should support disambiguation of homonymous terms. Ideally, increased precision would not come at the expense of recall.

Our second objective is to evaluate the contribution of different ConKG variants and KG embedding techniques in the ConLMs for the UVA task. More specifically, we investigate (1) which ConLM variant performs best; and (2) how the ConLMs compare to the reference LexLM.

Our last objective is to assess the performance of the ConLMs on various datasets containing pairs of terms with different degrees of lexical similarity. Because we expect contextual information to be useful mainly for disambiguation of homonymous terms, we hypothesize that performance improvement will be more markedly observed on datasets with a high degree of lexical similarity among negative examples. Ideally, increased performance on lexically similar terms would not come at the expense of the performance on less similar terms.

Contributions. Our contributions include:

- An approach to develop context-enriched learning models (ConLMs) with substantial performance improvement over the reference LexLM, namely +4.41% in precision (93.16%), +2.27% in F1 (92.88%), while retaining similar performance in recall.
- An extensive set of experiments for evaluating the ConLMs with the pairwise combination of ConKG variants and KG embedding techniques. Our experiments show that overall, the embedding technique TransE paired with the ConHR variant emerged as the best performer.
- An extensive set of experiments showing that the performance gain observed for the dataset with a high degree of lexical similarity among negative examples (+5.94% in precision, +1.49% in recall, and +3.72% in F1) is conserved, but to a lesser degree, on datasets with a lower degree of lexical similarity (between +0.5% and +1.06% in F1).

The remainder of the paper is organized as follows. Section 2 provides background knowledge about the Metathesaurus and how lexical and contextual information is transformed into ConKGs. Section 3 describes the KG embedding techniques selected for training the embedding vectors for the ConKGs. Section 4 describes how the ConLMs are developed. In Section 5, we present our exper-

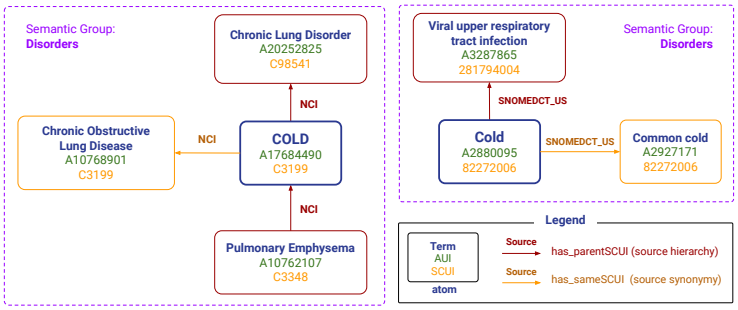

Fig. 1. Example illustrating the contextual information available for disambiguating the terms *COLD* from NCI and *Cold* from SNOMEDCT_US, including source synonyms (through **has_sameSCUI**), source semantic group (*Disorders*), and source hierarchical relations (through **has_parentSCUI**).

iments and their results. In Section 6, we discuss our findings and future work. In section 7, we discuss the related work. Section 8 concludes the paper.

2 Context-Enriched Knowledge Graphs

This section describes how multiple variants of context-enriched knowledge graphs (ConKG) are constructed using the various types of contextual information from source vocabularies. These ConKG variants will be used as input for training the KG embeddings in Section 3.

2.1 Background Knowledge on the UMLS Metathesaurus

Nguyen et al. [25] described the knowledge representation aspects of UMLS used in the synonymy prediction task. Here we briefly summarize key concepts from [25] and add new concepts specific to contextual information. The examples below are illustrated in Figure 1.

AUI. Every occurrence of a term in a source vocabulary is assigned a unique atom identifier (AUI). For example, "Cold" in SNOMEDCT_US and "COLD" in NCI are assigned different AUIs, "A2880095" and "A17684490", respectively.

SCUI and m_s. Each AUI is optionally associated with one identifier provided by its source ("Source CUI" or SCUI). Terms considered synonymous in a source vocabulary are assigned the same SCUI. For example, the terms "COLD" and "Chronic Obstructive Lung Disease" are associated with the same SCUI, "C3199", from the source vocabulary NCI. SCUIs play an important role in the Metathesaurus construction process because source synonymy is very often conserved in the Metathesaurus.

(M1) Let S be the set of SCUIs in the Metathesaurus. Let m_s be the function that maps an atom $a \in A$ to an SCUI $s \in S$ such that $s = m_s(a)$.

Source Semantic Group and m_g. Source semantic groups (SGs) are assigned to a source vocabulary (or to its top-level terms for multi-domain vocabularies). An SCUI from a source will inherit its semantic groups from its source.

For example, "COLD" with SCUI "C3199" inherits the SG "Disorders" from the top-level term "Disease, Disorder or Finding" (from NCI). Let G be the set of semantic groups in the Metathesaurus.

(M2) Let m_g be the function that maps an SCUI $s \in S$ to a set of semantic groups such that $m_g(s) \subset G$.

Source Hierarchical Relations and m_h. An SCUI may have parent or child terms in a source vocabulary. For example, "COLD" with SCUI "C3199" (from NCI) has "Chronic Lung Disorder" with SCUI "C98541" as a parent and "Pulmonary Emphysema" with SCUI "C3348" as a child.

(M3) Let m_h be the function that maps an SCUI $s \in S$ to a set of its parents, $m_h : S \rightarrow S$ such that $m_h(s) \subset S$.

2.2 Context-enriched Knowledge Graphs

Here we explain how we construct the context-enriched knowledge graphs (ConKGs) and define the set of triples constructed for each ConKG variant.

A is the set of AUIs, S is the set of SCUIs, G is the set of SGs, and the mapping functions $\{m_s, m_g, m_h\}$ are defined in M1, M2, and M3 above.

ConSS. Let r_s denote the binary relation has_SCUI from an AUI $a \in A$ to an SCUI $s \in S$. $\forall a \in A$, if $s = m_s(a) : (a, r_s, s) \in$ ConSS. The ConSS variant includes the triples representing the relationship between an AUI and its SCUI.

(V1) ConSS $= \{(a, r_s, s) : s = m_s(a)\}$.

ConSG. Let r_g denote the binary relation has_SG from an SCUI $s \in S$ to a SG $g \in G$. $\forall s \in S$, if $g \in m_g(s) : (s, r_g, g) \in$ ConSG. The ConSG variant includes the triples representing the relationship between an SCUI and its semantic groups.

(V2) ConSG $= \{(s, r_g, g) : g \in m_g(s)\}$.

ConHR. Let r_h denote the binary relation has_parentSCUI from an SCUI $s \in S$ to its parent SCUI $p \in S$. $\forall s \in S$, if $p \in m_h(s) : (s, r_h, p) \in$ ConHR. The ConHR variant includes the triples representing the relationship between an SCUI and its parent SCUI.

(V3) ConHR $= \{(s, r_h, p) : p \in m_h(s)\}$.

ConAll. ConAll = ConSS $\cup$ ConSG $\cup$ ConHR.

3 Embeddings for Context-enriched Knowledge Graphs

This section describes how the ConKG triples are transformed into their respective ConKG embedding vectors using various KG embedding techniques. We selected TransE [3], RESCAL [27], DistMult [34], HolE [26], and ComplEx [29] due to their demonstrated all-around performance. These trained ConKG embedding vectors will be then added to the LexLM to form the ConLMs in Section 4. A list of abbreviations is provided in Table 2 for convenience.

3.1 Knowledge Graph Embeddings

We explore different KG embedding approaches to transform the structural representation of ConKG triples T into a low-dimensional vector space, while preserving the semantics defined in the ConKG. Such transformation allows the ConKG triples to be added to the LexLM as a set of ConKG embedding vectors. Here we describe how ConKG triples are transformed into ConKG embedding vectors.

Given that A is the set of AUIs, S is the set of SCUIs, and G is the set of SGs, let E be the set of ConKG entities, $E = A \cup S \cup G$. Let R be the set of all ConKG relations, $R = \{r_s, r_g, r_h\}$. Let T be the set of ConKG triples, a triple $t \in T$ if $t = (e_1, r, e_2)$ with $r \in R$, and $e_1, e_2 \in E$. Let T' be the set of negative ConKG triples, $t' = (e'_1, r, e'_2) \in T'$ if $\exists t = (e_1, r, e_2) \in T$ and $t' \notin T$. Let $d = 2 * i$ ($i \in \mathbf{N}$) be the dimension of embedding vector. We choose d to be an even number to facilitate the representation of ComplEx vectors as explained later in this section.

Generating embedding vectors. KG embedding techniques generate the embeddings for entity and relation vectors of dimension d using a scoring function $f_r : E \times R \times E \rightarrow \mathbb{R}$. This scoring function measures the plausibility of facts by minimizing the loss function $L(T, T', \theta)$ with respect to parameter θ either by (a) distance-based (TransE) or (b) similarity-based functions (RESCAL, HolE, DistMult, and ComplEx). (See [13] for details about scoring and loss functions for the various embedding techniques.)

Entity embeddings. Let $\mathbf{E}$ be the set of embedding vectors of E, then $\mathbf{e} \in \mathbf{E}$ is an embedding vector of entity e. Here we only use entity embeddings because our ConKG has only three relations and these relation embeddings are not useful for our task. While the embedding for an entity or relation from TransE, HolE, and DistMult is a single vector with dimension $d = 2i$, ComplEx embeddings require two vectors (real and imaginary) of dimension $d = i$. In this case, we concatenate the real and imaginary vectors for each entity into a single vector of dimension $d = 2i$. For ComplEx, we denote the two-vector embeddings as $\mathbf{E} = (\mathbf{E}_{re}, \mathbf{E}_{im})$, then we define the embedding vector for an entity e as $\mathbf{E}(e) = \mathbf{E}_{re}(e) \oplus \mathbf{E}_{im}(e)\ \forall e \in \mathbf{E}$.

The output of each KG embedding technique is a set of entity embeddings: $\mathbf{E}_{ConSS}$, $\mathbf{E}_{ConSG}$, $\mathbf{E}_{ConHR}$, and $\mathbf{E}_{ConAll} \forall\, e \in E$, which will be used to derive the ConKG embeddings in the next section.

Table 1. Set of ConKG embeddings for each ConKG variant

Variant	ConKG Triples	Set of ConKG embedding vectors
ConSS	$\{(a, r_s, s) : s = m_s(a)\}$	$\{\mathbf{E}_{ConSS}(a) \oplus \mathbf{E}_{ConSS}(m_s(a)) : \forall a \in A\}$
ConSG	$\{(s, r_g, g) : g \in m_g(s)\}$	$\{\mathbf{E}_{ConSG}(m_s(a)) \oplus \sum_{j=1}^{\|m_g(m_s(a))\|} \mathbf{E}_{ConSG}(g_j) : \forall a \in A, g_j \in m_g(m_s(a))\}$
ConHR	$\{(s, r_h, p) : p \in m_h(s)\}$	$\{\mathbf{E}_{ConHR}(m_s(a)) : \forall a \in A\}$
ConAll	$\{(a, r_s, s) : s = m_s(a)\}$ $\{(s, r_g, g) : g \in m_g(s)\}$ $\{(s, r_h, p) : p \in m_h(s)\}$	$\{\mathbf{E}_{ConAll}(a) \oplus$ $\mathbf{E}_{ConAll}(m_s(a)) \oplus$ $\sum_{j=1}^{\|m_g(m_s(a))\|} \mathbf{E}_{ConAll}(g_j) : \forall a \in A, g_j \in m_g(m_s(a))\}$

Table 2. List of abbreviations used in the paper

Notion	Meaning	Notion	Meaning
AUI	Atom unique ID	A	set of $AUIs$
SCUI	Source concept unique identifier	T	set of ConKG triples
SS	Source synonym	T'	set of ConKG negative triples
SG	Semantic group	S	set of SCUIs
HR	Hierarchical Relation	m_s	$A \rightarrow S$
TOPN_SIM	Highest level of lexical similarity	G	set of SGs
RAN_NOSIM	Zero lexical similarity	m_g	$S \rightarrow G$
RAN_SIM	Low lexical similarity	m_h	$S \rightarrow S$
LexLM	Lexical-based learning model	E	$A \cup S \cup G$
ConLM	Context-enriched learning model	$\mathbf{E}$	set of entity embeddings
ConKG	Context-enriched knowledge graph	$\mathbf{C}$	set of ConKG embeddings
TRAIN_	Prefix of training datasets	$\sum$	average an array of vectors
GEN_	Prefix for generalization test sets	$\oplus$	concatenate vectors

3.2 ConKG Embeddings

We derive ConKG embeddings $\mathbf{C}$ for each type of contextual information described in Section 2.2 with respect to $a \in A$, so that we can add them to the word embedding vectors from the LexLM described in Section 4. For each type of contextual information, we generate a ConKG embedding vector $\mathbf{c} \in \mathbf{C}$ for each $a \in A$ by concatenating the entity embeddings of ConKG entities inside the ConKG triples corresponding to a, including $m_s(a)$ for an SCUI, and $m_g(m_s(a))$ for a semantic group. As an SCUI is mapped to a set of semantic groups, we get the entity embedding for each semantic group and average the set of embedding vectors ($\sum$). We reuse the mapping functions defined in Section 2.2. Table 1 shows the set of the ConKG embeddings for each type of contextual information. The output of each KG embedding technique is a set of context-enriched embeddings for all AUIs: $\mathbf{C}_{ConSS}$, $\mathbf{C}_{ConSG}$, $\mathbf{C}_{ConHR}$, and $\mathbf{C}_{ConAll}$, $\forall a \in A$, which will be added to the reference LexLM in Section 4 and evaluated in Section 5.

4 Neural Network Architecture

This section describes the LexLM architecture from [25], which we will be using as the reference model, and our approach in adding the context-enriched embeddings from Section 3 to the LexLM.

LexLM. The LexLM (grey boxes in Figure 2) adopts the Siamese architecture [23] that takes in a pair of inputs and learns representations based on explicit similarity and dissimilarity information defined during training. The inputs (a pair of atoms) are pre-processed and transformed into their numerical representations with BioWordVec embeddings pre-trained from PubMed text corpus and MeSH data [35]. The word embeddings are then fed to Long Short Term Memory (LSTM) layers to learn the semantic and syntactic features of the atoms through time. The LexLM relies exclusively on the lexical features of the atoms, i.e., the terms themselves.

ConLMs. We develop the ConLM (Figure 2), which adds to the LexLM (at the LSTM layer) the different variants of ConKGs described in Section 3. For each ConKG variant, we first feed the respective trained ConKG embedding

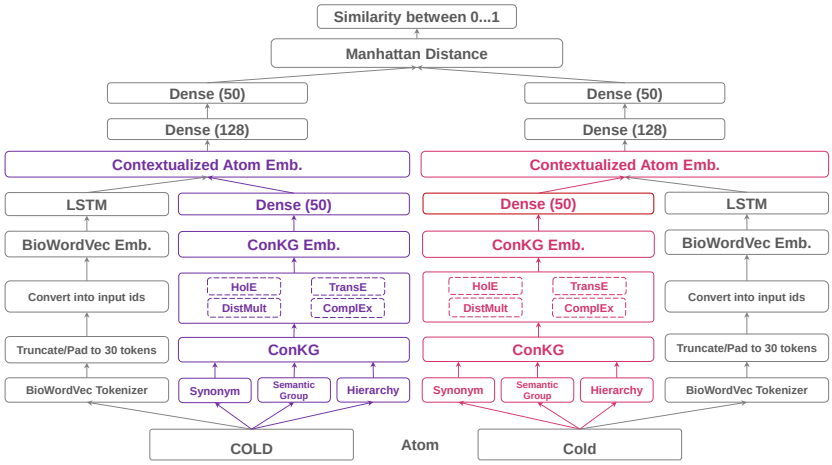

Fig. 2. The architecture of the neural network in the Context-enriched Learning Model (ConLM) created by adding the ConKG embeddings to the Lexical-based Learning Model (LexLM) embeddings (grey boxes) from [25].

vectors to a 50-unit dense layer to learn the derived features in Table 1. We then concatenate $\oplus$ the output of the dense layer with the output of the LSTM units from the LexLM. Together, they form the contextualized atom embedding, which is then fed to subsequent dense layers with 128 and 50 learning units respectively. The output is a Manhattan distance similarity function [1], which computes a score indicating the degree of synonymy between the atoms. The datasets used for training and testing the ConLMs are described in Section 5.2. The trained models are evaluated to assess their respective contribution in Section 5.

5 Evaluation

This section presents a set of experiments to evaluate the hypotheses described in Section 1. The experiments are reproducible with the materials temporarily available for reviews at https://bit.ly/iswc-supp-4 with a no-cost UMLS license [1].

5.1 Experimental Setup

We conducted three types of experiments: (1) training embeddings for ConKG variants using the different KG embedding techniques described in Section 5.3 and comparing their performance against that of the reference LexLM, (2) training and testing ConLMs using ConKG variants in Section 5.4 and 5.5 respectively to identify the best combination, and (3) testing the combination of ConKG variants with lexical similarity variants in Section 5.6 (with the best KG embedding technique obtained from Section 5.5) to demonstrate that contextual information mostly benefits datasets with high levels of lexical similarity.

[1] https://uts.nlm.nih.gov/uts/

Table 3. The ConKG variants with their respective number of unique entities, relations, positive and negative triples

ConKG	# of Entities E	# of Relations R	# of Positive Triples	# of Negative Triples
ConAll	12,366,913	3	16,425,157	16,425,157
ConSS	12,366,913	3	8,713,194	8,713,194
ConHR	12,366,913	3	3,520,969	3,520,969
ConSG	12,366,913	3	4,190,994	4,190,994

All these experiments are deployed as batches of parallel jobs with the Slurm [2] workload manager to the Biowulf high-performance computing cluster [3] at the National Institutes of Health (NIH). We used Tesla V100x GPUs with 32GB of GPU RAM and at least 220GB of CPU RAM for each training and testing task. While the implementation is configurable and reproducible in a different environment, these experiments are computationally- and resource-intensive. We estimated that we used over 2500 GPU hours for the set of experiments reported in this paper. Run-time information is provided in 5.3 and 5.4.

5.2 Datasets

This section describes the datasets for the three types of experiments. We used release 2020AA of the UMLS Metathesaurus restricted to English terms from active source vocabularies. The UMLS can be downloaded with a no-cost license [1].

Datasets for Training Embeddings for ConKG Variants. Table 3 shows the characteristics of the four datasets generated for training the KG embeddings for each ConKG variant. We use a 1:1 positive to negative triple ratio for generating the triple instances described in Section 2.2. The negative triples are automatically generated using the "bern" sampling technique [32] to corrupt either the e_1 or e_2 entity.

Datasets for Training and Testing ConLMs. To compare the ConLMs against the LexLM baseline, we reuse the training and testing datasets generated in [25]. More specifically, the LexLM variant trained on the TRAIN_ALL dataset performs best across the four generalization tests (GEN_ALL, GEN_TOPN_SIM, GEN_RAN_SIM, and GEN_RAN_NOSIM) and is used as baseline here. In practice, we use the dataset TRAIN_ALL for training our ConLMs (Section 5.4), and the four generalization datasets for testing our ConLMs (Section 5.5 and 5.6).

5.3 Training 16 Variants of ConKG Embeddings

Training parameters. We use the OpenKE [4] library [10] to implement the KG embedding techniques to train the embeddings independently for each ConKG variant described in Section 2.2. Since this is not a systematic evaluation, nor a performance benchmark across various KG embedding techniques, we did not

[2] https://slurm.schedmd.com/documentation.html
[3] https://hpc.nih.gov/
[4] https://github.com/thunlp/OpenKE

Table 4. Training results for each combination of ConKG variant (ConSS for source synonymy, ConSG for semantic group, ConHR for hierarchical relations, and ConAll for all of them) and KG embedding technique (TransE, HolE, DistMult, and ComplEx), with loss (final epoch) and time (seconds per epoch).

Model	ConAll		ConSS		ConHR		ConSG	
	loss	time	loss	time	loss	time	loss	time
DistMult	0.1191	20.087	0.0418	13.243	0.2803	8.772	0.0088	9.049
HolE	4326582.2109	23.696	3550789.6875	13.235	631802.8467	8.975	269649.3496	9.001
ComplEx	0.3766	31.612	0.1380	14.414	0.2129	7.473	0.0207	9.418
TransE	0.0054	4.000	0.0013	2.025	0.0013	0.808	0.0009	0.972

select the list of hyper-parameters that work best for each technique. Instead we ran various hyper-parameter selection experiments and finalized a list of hyper-parameters that balance performance and training speed, as well as maximize the GPU memory across all techniques. Each ConKG variant is trained with each KG embedding technique with (a) 1000 epochs, (b) batch size of 50, (c) learning rate of 0.05, (d) loss margin of 1.0, (e) 1:1 positive to negative triple sampling ratio, and with (f) embedding dimension of size 100. In terms of learning optimizer, we found that stochastic gradient descent (SGD) worked best for TransE and Adam [16] for HolE, DistMult, and ComplEx. The full (1000 epochs) training for each combination of ConKG variant and KG embedding technique takes approximately nine hours. Table 4 shows the training results.

In our training, we observed that RESCAL was not able to converge (despite hyper-parameters tuning). Therefore, we omit reporting RESCAL results in Table 4 and exclude it from experiments (2) and (3) in Section 5.1.

5.4 Training and Testing 16 ConLM Variants

We trained 16 ConLM variants using the 16 variants of the ConKG embeddings obtained from the training presented in Section 5.3. We use the TRAIN_ALL dataset variant from the training of the LexLM in [25] for training our ConLMs. We also use the same training hyper-parameters from the LexLM, namely 100 epochs, batch size of 8192, and Adam optimizer. Training and predicting each epoch of the ConLMs takes about 30 minutes.

We test these 16 trained ConLM variants using the GEN_ALL generalization test set. We report the results in Table 5 and discuss them in Section 5.5.

Since TransE performs best all-around, we test the four trained ConLMs based on TransE using the generalization test sets (GEN_ALL, GEN_TOPN_SIM, GEN_RAN_SIM, and GEN_RAN_NOSIM). We report the results in Table 6 and discuss them in Section 5.6.

5.5 Comparing pairwise combinations of ConKG variants and KG embedding techniques

We have trained the ConLMs with the TRAIN_ALL dataset and tested them with the GEN_ALL test set (as discussed in Section 5.4). Here we discuss the

Table 5. Experimental results for each combination of ConKG variant (ConSS for source synonymy, ConSG for semantic group, ConHR for hierarchical relations, and ConAll for all of them) and KG embedding technique (TransE, HolE, DistMult, and ComplEx). Each combination is also compared against the baseline LexLM [25]. The difference between the TransE ConKG variant and the LexLM baseline is shown in the row TransE-LexLM. The RBA baseline is provided for reference.

	ConAll				ConSS			
	accuracy	precision	recall	F1	accuracy	precision	recall	F1
DistMult	0.9946	0.9094	0.9250	0.9172	0.9879	**0.9339**	0.6739	0.7828
HolE	0.9869	0.9064	0.6643	0.7667	0.9945	0.9200	0.9102	0.9151
ComplEx	0.9943	0.9068	0.9196	0.9131	**0.9951**	0.9293	0.9180	**0.9236**
TransE	**0.9949**	**0.9156**	**0.9277**	**0.9216**	0.9905	0.9247	**0.9204**	0.9226
LexLM	0.9938	0.8875	0.9254	0.9061	0.9938	0.8875	0.9254	0.9061
TransE-LexLM	0.0011	0.0281	0.0023	0.0155	0.0012	0.0372	-0.0050	0.0165
RBA	0.9863	0.8631	0.6871	0.7651	0.9863	0.8631	0.6871	0.7651
	ConHR				ConSG			
	accuracy	precision	recall	F1	accuracy	precision	recall	F1
DistMult	0.9859	0.8711	0.6640	0.7536	0.9946	0.9101	0.9258	0.9179
HolE	0.9858	0.8688	0.6639	0.7526	0.9869	0.9069	0.6641	0.7668
ComplEx	0.9850	0.8481	0.6567	0.7402	0.9951	**0.9208**	0.9286	0.9247
TransE	**0.9954**	**0.9316**	**0.9260**	**0.9288**	**0.9952**	0.9191	**0.9346**	**0.9268**
LexLM	0.9938	0.8875	0.9254	0.9061	0.9938	0.8875	0.9254	0.9061
TransE-LexLM	**0.0016**	**0.0441**	**0.0006**	**0.0227**	0.0014	0.0316	0.0092	0.0207
RBA	0.9863	0.8631	0.6871	0.7651	0.9863	0.8631	0.6871	0.7651

test results obtained from these ConLM variants with different combinations of KG embedding techniques and ConKG variants.

Comparing KG embedding techniques and ConKG variants. Table 5 shows that the TransE embedding technique paired with the ConHR variant performed best overall, with 99.54% accuracy, 93.16% precision, 92.59% recall, and 92.88% F1. Compared to the best LexLM variant from [25], the TransE-ConHR combination improved 0.16% in accuracy, 4.41% in precision, and 2.27% in F1, while retaining similar performance in recall. This result validates our hypothesis that the addition of contextual information to the LexLM would improve performance. Of note, several ConLM variants outperformed the reference LexLM (e.g., TransE paired with any ConKG variant, ComplEx paired with ConAll, ConSS, or ConSG.) The performance of the poorly performing ConLMs is similar to that of the RBA.

Comparing KG embedding techniques. In Table 5, TransE scored highest in every measure for the three variants (ConAll, ConSG, and ConHR), while ComplEx and TransE performed similarly for the ConSS variant. ComplEx scored second highest with a close performance to TransE in the three variants ConAll, ConSS, and ConSG, but severely suffered from both low precision and very low recall when learning hierarchical relations in the ConHR variant. DistMult performed well for the ConAll and ConSG variants, but similar to

Table 6. Experimental results for each combination of ConKG variant (ConSS for source synonymy, ConSG for semantic group, ConHR for hierarchical relations, and ConAll for all of them) and degree of lexical similarity (GEN_ALL, GEN_TOPN_SIM, GEN_RAN_SIM, and GEN_RAN_NOSIM). Each combination is trained with the same TRAIN_ALL dataset, and compared against the LexLM baseline [25]. Differences from the baseline are shown in the *Diff* rows.

		GEN_ALL				GEN_TOPN_SIM			
		accuracy	precision	recall	F1	accuracy	precision	recall	F1
ConAll	Score	0.9949	0.9156	**0.9277**	0.9216	0.9869	0.9253	**0.9333**	0.9293
	Diff	0.0011	0.0281	0.0023	0.0155	0.0062	0.0412	0.0223	0.0319
ConSS	Score	0.9950	0.9247	0.9204	0.9226	0.9860	0.9342	0.9136	0.9237
	Diff	0.0012	0.0372	-0.0050	0.0165	0.0053	0.0501	0.0026	0.0263
ConHR	Score	**0.9954**	**0.9316**	0.9260	**0.9288**	**0.9880**	**0.9435**	0.9259	**0.9346**
	Diff	0.0016	0.0441	0.0006	0.0227	0.0073	0.0594	0.0149	0.0372
ConSG	Score	0.9953	0.9203	0.9359	0.9280	0.9879	0.9333	0.9359	**0.9346**
	Diff	0.0015	0.0328	0.0105	0.0219	0.0072	0.0492	0.0249	0.0372
LexLM	Score	0.9938	0.8875	0.9254	0.9061	0.9807	0.8841	0.9110	0.8974
		GEN_RAN_SIM				GEN_RAN_NOSIM			
		accuracy	precision	recall	F1	accuracy	precision	recall	F1
ConAll	Score	0.9922	0.9822	**0.9333**	0.9571	0.9938	0.9955	0.9333	0.9634
	Diff	0.0017	-0.0036	0.0223	0.0102	0.0014	-0.0016	0.0171	0.0085
ConSS	Score	0.9909	0.9872	0.9136	0.9490	0.9922	0.9943	0.9157	0.9534
	Diff	0.0004	0.0014	0.0026	0.0021	-0.0002	-0.0028	-0.0005	-0.0015
ConHR	Score	0.9924	**0.9912**	0.9259	0.9575	0.9932	0.9964	0.9259	0.9599
	Diff	0.0019	0.0054	0.0149	0.0106	0.0008	-0.0007	0.0097	0.0050
ConSG	Score	**0.9929**	0.9869	**0.9359**	**0.9607**	**0.9942**	**0.9972**	**0.9359**	**0.9656**
	Diff	0.0024	0.0011	0.0249	0.0138	0.0018	0.0001	0.0197	0.0107
LexLM	Score	0.9905	0.9858	0.9110	0.9469	0.9924	0.9971	0.9162	0.9549

ComplEx, suffered from very low recall in the ConSS and ConHR variants. HolE only performed well with the ConAll variant.

Comparing ConKG variants. Although ConHR paired with TransE performed best, the three other embedding techniques failed to learn from it. Each of the three variants, ConAll, ConSS, and ConSG, failed to learn with one embedding technique. In other words, the learning in these three variants is more stable than the ConHR variant across all embedding techniques.

5.6 Comparing ConKG variants across different generalization tests using the best KG embedding technique

We trained the ConLMs with the TRAIN_ALL dataset and tested them with the four generalization tests (as described in Section 5.4) using TransE (i.e., the best KG embedding technique, as shown in Section 5.5). Here we discuss the results obtained from the ConLMs variants for different ConKG variants.

Comparing ConKG variants. Table 6 shows that the ConHR and ConSG variants consistently outperformed the LexLM baseline, with performance gains in every measure. The ConHR variant performed best for the tests GEN_ALL and GEN_TOPN_SIM, while the ConSG performed best for the tests GEN_RAN_SIM and GEN_RAN_NOSIM. The ConAll variant performed best in recall among the ConKG variants for the tests GEN_ALL, GEN_TOPN_SIM.

Comparing generalization tests. As shown in the **Diff** rows in Table 6, adding hierarchical relations yielded the highest gain in precision (+5.94%), recall (+1.49%), and F1 (+3.72%) with the GEN_TOPN_SIM dataset, i.e., the dataset with the highest level of lexical similarity among negative examples. This result validates our hypothesis that the addition of contextual information to the LexLM would improve performance more markedly on lexically similar terms. Of note, it also improved the performance in generalization tests with lower levels of lexical similarity: GEN_RAN_SIM (+0.54% in precision, +1.49% in recall, and +1.06 in F1) and GEN_RAN_NOSIM (-0.07% in precision, 0.97% in recall, and 0.5% in F1), albeit more modestly. Interestingly, increased performance on lexically similar terms has not negatively affected the performance on less similar terms. In fact, we observed some performance (in F1) on all test datasets.

6 Discussion and Future Work

Findings. As we hypothesized, the use of KG embeddings to add contextual information to the reference LexLM yielded substantial performance improvement over the LexLM baseline, especially in terms of precision. Even more remarkably, there was no concomitant loss of recall and the overall performance (F1) also increased. We showed a large degree of variability among KG embedding techniques and among types of contextual information. TransE performed best overall, especially in combination with hierarchical relations. Finally, our experiments also confirmed that the addition of contextual information not only benefits pairs of terms with high lexical similarity as we expected, but also improves performance on all test datasets.

Significance. The performance of the reference LexLM was already very strong, especially given the limited information available to this model (i.e., only the lexical features of terms). However, given the very large number of comparisons required for the Metathesaurus construction, the number of false positives remained important despite precision values near 90%. Therefore, increasing precision by 4.41% (to 93.16%) represents a substantial advance for the UMLS vocabulary alignment (UVA) task, especially because recall was not negatively affected. Another important result for the application of ConLMs to the UVA task is that the addition of contextual information to the LexLM improves performance on all datasets, not only on highly similar terms. This shows that the performance of ConLMs will likely generalize to the UMLS Metathesaurus as a whole.

Limitation and Future Work. As mentioned earlier, this investigation is no substitute for a systematic performance evaluation of KG embedding tech-

niques and thorough benchmarking. Our focus was rather on the application to the UVA task. Performing a comprehensive analysis of the differences with the LexLM baseline was beyond the scope of this investigation, but is part of our future work. Finally, we are also planning to improve the performance of the LexLM, e.g., by developing novel embedding techniques that consider the unique characteristics of the UMLS Metathesaurus (especially the specific lexical features and hierarchical organization of biomedical terms).

7 Related Work

Biomedical ontology alignment is a long-standing research effort driven by the Ontology Alignment Evaluation Initiative (OAEI) since 2005. With the growth of interest in integrating biomedical ontologies at scale [25], studies have looked into using rule-based and statistical approaches [8, 24, 14], as well as supervised learning approaches for ontologies matching [7] by assessing the similarity [18, 30] and relatedness [19] between words and sentences. Such tasks are also known as Semantic Text Similarity (STS) tasks. Recent progress has been attributed to the use of a combination of knowledge-based similarity with deep learning techniques, such as word embeddings [20] for input feature representation, and Siamese Network [23, 11, 2, 12] to learn the underlying semantics and structure. Unsupervised approaches, such as transformer-based mechanisms, have also been explored with a great degree of success [6]. Nonetheless, these techniques are largely based on lexical features and require very large text corpora to learn from. In our approach, we also exploit the graph structure of various types of contextual information through the use of KG embeddings.

There are several families of KG embedding techniques [31]. Since this is the first attempt to use the various types of contextual information in the UVA task at scale, we explored two of the popular classes of techniques because of their demonstrated success in various tasks: transitional distance-based with TransE [3], and semantic matching-based using RESCAL [27], DistMult [34], HolE [26], and ComplEx [29] [31]. We evaluated their performance in the UVA task, but did not benchmark them against other forms of graph representation techniques [13, 33], such as Graph Convolutional Networks (GCNs) [17].

8 Conclusion

In summary, we demonstrated the importance of using contextual information in the UMLS vocabulary alignment. More specifically, we showed that it was possible to improve on the performance of a learning model based on the lexical features of biomedical terms by taking into account the context of these terms in their source vocabulary (source synonymy, source semantics, hierarchical relations). Adding contextual information to the lexical model through KG embeddings yielded a substantial gain in precision with no negative effect on recall and was particularly beneficial to lexically similar strings, such as homonyms, but also improved performance overall.

References

1. C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.
2. A. Bento, A. Zouaq, and M. Gagnon. Ontology matching using convolutional neural networks. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5648–5653, 2020.
3. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9, 2013.
4. J. J. Cimino. Auditing the unified medical language system with semantic methods. *Journal of the American Medical Informatics Association*, 5(1):41–51, 1998.
5. J. J. Cimino, H. Min, and Y. Perl. Consistency across the hierarchies of the umls semantic network and metathesaurus. *Journal of biomedical informatics*, 36(6):450–461, 2003.
6. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
7. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology matching: A machine learning approach. In *Handbook on ontologies*, pages 385–403. Springer, 2004.
8. D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto. The agreementmakerlight ontology matching system. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 527–541. Springer, 2013.
9. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.
10. X. Han, S. Cao, L. Xin, Y. Lin, Z. Liu, M. Sun, and J. Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.
11. H. He, K. Gimpel, and J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1576–1586, 2015.
12. D. Huang, A. Ahmed, S. Y. Arafat, K. I. Rashid, Q. Abbas, and F. Ren. Sentence-embedding and similarity via hybrid bidirectional-lstm and cnn utilizing weighted-pooling attention. *IEICE TRANSACTIONS on Information and Systems*, 103(10):2216–2227, 2020.
13. S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020.
14. E. Jiménez-Ruiz and B. C. Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288, 2011.
15. A. Jimeno-Yepes, E. Jiménez-Ruiz, R. Berlanga-Llavori, and D. Rebholz-Schuhmann. Reuse of terminological resources for efficient ontological engineering in life sciences. *BMC bioinformatics*, 10(S10):S4, 2009.
16. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
17. T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

18. P. Kolyvakis, A. Kalousis, B. Smith, and D. Kiritsis. Biomedical ontology alignment: an approach based on representation learning. *Journal of biomedical semantics*, 9(1):1–20, 2018.
19. Y. Mao and K. W. Fung. Use of word and graph embedding to measure semantic relatedness between unified medical language system concepts. *Journal of the American Medical Informatics Association*, 2020.
20. R. Mihalcea, C. Corley, C. Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780, 2006.
21. C. P. Morrey, J. Geller, M. Halper, and Y. Perl. The neighborhood auditing tool: a hybrid interface for auditing the umls. *Journal of biomedical informatics*, 42(3):468–489, 2009.
22. F. Mougin, O. Bodenreider, and A. Burgun. Analyzing polysemous concepts from a clinical perspective: Application to auditing concept categorization in the umls. *Journal of Biomedical Informatics*, 42(3):440–451, 2009.
23. J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
24. D. Ngo and Z. Bellahsene. Overview of yam++?(not) yet another matcher for ontology alignment task. *Journal of Web Semantics*, 41:30–49, 2016.
25. V. Nguyen, H.-Y. Yip, and O. Bodenreider. Biomedical vocabulary alignment at scale in the umls. In *The Web Conference (WWW2021)*. ACM, 2021.
26. M. Nickel, L. Rosasco, and T. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
27. M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011.
28. A. Sharma, P. Talukdar, et al. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 122–131, 2018.
29. T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080. PMLR, 2016.
30. L. L. Wang, C. Bhagavatula, M. Neumann, K. Lo, C. Wilhelm, and W. Ammar. Ontology alignment in the biomedical domain using entity definitions and context. *arXiv preprint arXiv:1806.07976*, 2018.
31. Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.
32. Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.
33. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.
34. B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint 1412.6575*, 2014.
35. Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):1–9, 2019.